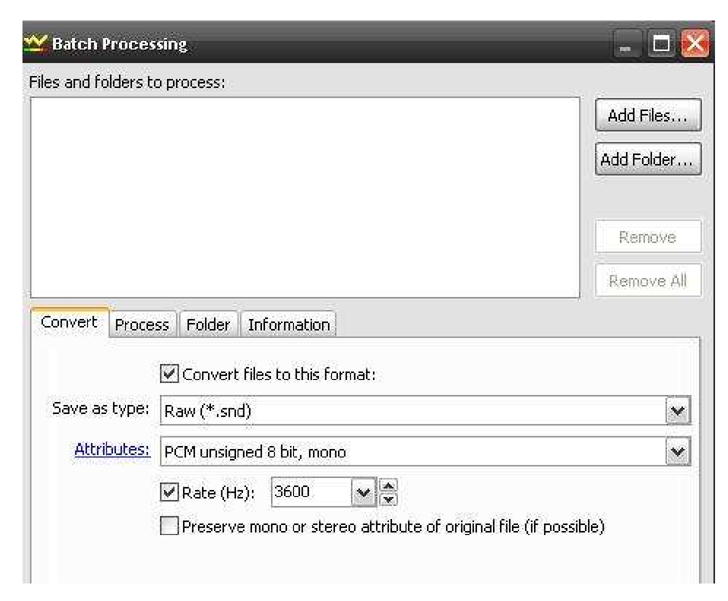
PROYECTO DE NAVIDAD
PIC QUE REPRODUCE SONIDO
Manolo Romero 2015

VISIÓN GENERAL

El circuito está compuesto por una EEPROM serie con bus tipo I2C modelo 24LC256 y un PIC tipo 16F628, un DAC de 5 Bit con resistencias ponderadas y un amplificador de audio con transistor en configuración de “seguidor de emisor”.
Cuando alimentamos el circuito el microcontrolador PIC lee en bucle todo el contenido de la EEPROM y manda los bytes leídos al puerto B donde serán transformados de nuevo en audio por el DAC y una vez amplificados son enviados al altavoz..

CODIFICACIÓN DEL AUDIO, LA EEPROM

El proyecto comienza con un mensaje de audio de pocos segundos de duración, el mensaje es una voz femenina que pronuncia “Feliz navidad”.
El audio original es un fichero tipo .WAV que vamos a codificar de manera adecuada para poder grabarlo en la EEPROM.
He utilizado el programa gratuito GOLDWAVE, hay versiones de pago y muchas otras gratuitas que hacen la misma función.



Utilizando la aplicación de “Batch processing” (procesado por lotes) vamos a convertir el fichero mensaje.wav en un fichero tipo PCM de 8 bit mono muestreado a 3600Hz.
El fichero resultante tiene la extensión .SND, procedo entonces a cambiar la extensión del fichero desde una ventana de DOS, y renombro el fichero como “mensaje.bin”.
Éste fichero ,.BIN es binario puro y puede grabarse directamente en le EEPROM 24LC256 usando un programador/grabador adecuado.

La EEPROM 24LC256 tiene 256Kbits, es decir 256x1024 bits = 262144 bits en total.
Si dividimos 262144 bits entre 8 bit, tenemos 32768 Bytes.
El audio ha sido muestreado a 3600Hz, por tanto para reproducirlo empleamos la misma velocidad de lectura, así que accedemos a 3600 bytes por segundo.
Por tanto si hacemos la división, tenemos 32768/3600=9,1 segundos.
En resumen, redondeando en la EEPROM caben 9 segundos de audio.
Utilizando éste método podemos grabar cualquier sonido de audio en la EEPROM.
También podemos grabar tonos o señales siempre que sean de baja frecuencia, la calidad no puede ser buena con este ancho de banda tan pequeño, el sonido es parecido a una señal telefónica.

Trucos:

Procure filtrar el fichero mensaje.wav antes de procesarlo, en concreto es interesante pasarle un filtro pasa bajos para eliminar alta frecuencia, además recomiendo maximizar su volumen pues el amplificador de salida no amplifica tensión solo amplifica corriente para atacar la baja impedancia del altavoz (8 ohmios) .

FUNCIÓN DEL PIC

El PIC está programado en ensamblador, al final de éste documento pueden ver el código fuente (no compilable), no incluyo las librerías pues son privadas y las sigo usando, son librerías sencillas que pueden programar ustedes, no realizan funciones especiales ni extrañas, además explicaré brevemente que hace cada una de ellas.

El PIC en realidad hace un bucle en modo infinito, se limita a leer sucesivamente todos los bytes de la EEPROM y los va mandando al puerto B.
El proceso es el siguiente:
Inicializa EEPROM +lee primer byte+manda byte a PORTB+lee segundo byte+manda byte a PORTB+lee tercer byte+manda byte a PORTB………..etc
Al llegar al final la EEPROM se desborda y pone su contador interno a cero, por tanto vuelve al principio, mientras el PIC sigue con su bucle.
Este proceso se realiza a una velocidad de 3600 bytes por segundo.

EL DAC

El convertidor Digital-Analógico es de 5 Bit, el utilizar 8 Bit proporciona mejor calidad, pero dado el escaso ancho de banda la mejora se nota poco.
Es un convertidor de tipo resistivo con resistencia ponderadas la relación entre una y la siguiente es siempre el doble.
Por temas de sencillez y coste no he utilizado resistencias de precisión así que empleo los valores más corrientes de la serie E24, si lo desean pueden utilizar un DAC de otro tipo (incluso integrado) o resistencias con valores más precisos.

EL AMPLIFICADOR

Como dije en líneas anteriores, conviene digitalizar el audio de la EEPROM con el máximo volumen posible sin recortar (distorsión) las puntas.
El amplificador de salida es del tipo Buffer de corriente, en concreto es una simple etapa de colector común (seguidor de emisor).
Así que ésta etapa no amplifica tensión, solo amplifica corriente para adaptar la impedancia de salida del DAC a la baja impedancia del altavoz (8 ohmios).
En el diseño he añadido un preset que permite un ajuste del volumen de salida.

UTILIDADE S DEL CIRCUITO

Son muchas tantas como a usted se le ocurra:
-Mensajes grabados para un contestador automático.
-Mensajes automáticos de alarma para megafonía:
-Sonidos de animales para pequeños juguetes.
-Generadores de señal de baja frecuencia
-Timbre o avisador con mensaje grabado
Etc…etc….etc
Ejemplo un sensor de radiación al pasar cierto nivel puede disparar el mensaje:
"Alarma nivel de radiación peligroso, abandone la zona"
Por último cabe destacar el aspecto didáctico del circuito.

LIMITACIONES Y/O MEJORAS.

Al comienzo del proyecto me preguntaron si el circuito puede grabar audio en tiempo real, la respuesta fue, NO, el tiempo de grabación de la Flash es de 2msg lo cual deja un ancho de banda insuficiente para un mensaje de audio mínimamente entendible.
Por tanto no es posible usar un PIC con ADC para grabar la EEPROM pues no tenemos suficiente velocidad en la misma para grabar audio en tiempo real.
El ancho de banda es limitado, así que la calidad del sonido no es alta y siempre vamos a perder las frecuencias altas de la señal al convertirla y grabarla en la EEPROM.
He simplificado el diseño todo lo posible, así que el audio no tiene toda la calida que se puede obtener, puede mejorarse, pero no esperen alta fidelidad, la calidad será siempre del tipo "telefónico"
Las mejoras serían:

- Usar un cristal de 4Mhz en vez del oscilador interno RC
- Usar un DAC integrado o resistencias de precisión.
- Usar un amplificador de audio tipo LM386 (integrado)
- Filtrar la salida del DAC para filtrar los escalones.

En algunas EEPROM (no en todas) he notado que al pasar de una página de memoria a otra, suelen tardar más o menos tiempo (depende del modelo) esto a veces origina un sonido como "golpeteo" en el audio, creo que es porque estoy utilizando la EEPROM LC (bajo consumo) y su tiempo de acceso a página es mayor.
Por tanto pueden probar otros modelos y ver si este pequeño efecto se reduce.

CÓDIGO FUENTE DEL PIC

A continuación pueden ver el código del PIC, el cual está realizado en lenguaje ensamblador: (luego lo tienen comentado)

```
;-------------------------------------------------------------------------------------------
;                              programa para que hable el pic
;                                        PIC 16F628
;                                © Manolo Romero 2015


        LIST P=16f628  ;Selección de micro

    __CONFIG _CP_OFF & _BODEN_OFF & _MCLRE_OFF & _WDT_OFF &
_PWRTE_ON & _INTRC_OSC_NOCLKOUT & _LVP_OFF & DATA_CP_OFF


    ERRORLEVEL -302 ;Quita los molestos warnings del compilador

     CBLOCK 20h
     PAGINA
     ENDC

        ORG    0
        goto    comienzo


;Configuración de puertos
comienzo
        BCF    STATUS,RP1
        BSF    STATUS,RP0 ;Selección de página 1
        MOVLW b'00000000' ;Configuramos puerto B salida
        MOVWF TRISB ;Cargamos TRISB
        CLRF PORTA ;borro PORTA
        MOVLW 0x07 ; desactivo el comparador interno del PIC
        MOVWF CMCON ;Activo los pin como I/O
        MOVLW b'11011000'
        MOVWF TRISA ;pongo RA<4:3:6:7> como inputs
        BCF    STATUS,RP0 ;Selección de página 0
        CLRF PORTB  ;Borro puerto B
        CLRF PORTA  ;Borro puerto A
        CLRF PAGINA
        CLRF 24LC256_H ;página0
vuelta  CLRF 24LC256_L  ; posición 0
        CALL 24LC256_inicializa
        CALL I2C_lee_byte
        MOVWF PORTB
        INCF PAGINA,1
        BTFSS STATUS,Z
        GOTO vuelta
```

```
 INCF 24LC256_H,1 ;página0
 BCF STATUS,Z
 CLRF PORTB
 GOTO vuelta
; LIBRERIAS
INCLUDE "p16f628.inc"
INCLUDE <BUS_I2C_.INC>
INCLUDE <24LC256.INC>

    END
```

Como pueden ver es un programa muy sencillo, hay poco que explicar si ustedes han programado PIC en ensamblador pueden ver lo que hace el programa.
De todas formas, dado que es muy reducido, vuelvo a poner todo el código, y añado los comentarios oportunos sin añadir los molestos “;” y explico línea a línea lo que creo que es más oportuno:

```
;-----------------------------------------------------------------------------------------------
;                              programa para que hable el pic
;                                        PIC 16F628 (debe funcionar con la versión A)
;                                © Manolo Romero 2015


   LIST P=16f628  ;Selección de micro (nada que añadir ☺)

   __CONFIG _CP_OFF & _BODEN_OFF & _MCLRE_OFF & _WDT_OFF &
_PWRTE_ON & _INTRC_OSC_NOCLKOUT & _LVP_OFF & DATA_CP_OFF
 (Éstos son los fusibles o bit de configuración para aquellos grabadores que los
reconocen en el fichero.hex)

   ERRORLEVEL -302 ;Quita los molestos warnings del compilador (Exacto)

    CBLOCK 20h (comienzo del bloque de variables del pic 16F628)
    PAGINA        (defino una variable PAGINA)
    ENDC           (FIN del bloque de variables

      ORG   0        (comienzo desde el reset será comienzo)
      goto    comienzo   (Saltamos)


;Configuración de puertos
comienzo
      BCF    STATUS,RP1
      BSF    STATUS,RP0 ;Selección de página 1
      MOVLW b'00000000' ;Configuramos puerto B salida
      MOVWF TRISB ;Cargamos TRISB
      CLRF PORTA ;borro PORTA
      MOVLW 0x07 ; desactivo el comparador interno del PIC
      MOVWF CMCON ;Activo los pin como I/O
```

```
          MOVLW b'11011000'
          MOVWF TRISA ;pongo RA<4:3:6:7> como inputs
          BCF    STATUS,RP0 ;Selección de página 0
          CLRF PORTB  ;Borro puerto B
          CLRF PORTA  ;Borro puerto A
          CLRF PAGINA
          CLRF 24LC256_H ;página0
vuelta    CLRF 24LC256_L  ; posición 0
          CALL 24LC256_inicializa     (llamada inicializa la EEPROM)
          CALL I2C_lee_byte           (llamada lee un byte)
          MOVWF PORTB                 (el byte leido lo mando al puerto B)
          INCF PAGINA,1
          BTFSS STATUS,Z              (el byte no es último vuelta al bucle)
          GOTO vuelta
          INCF 24LC256_H,1 ;página0   (incrementamos el contador de página)
          BCF STATUS,Z                (borro el flag Z)
          CLRF PORTB                  (borro el puerto)
          GOTO vuelta                 (vuelta a comenzar, bucle infinito ☺)
        ; LIBRERIAS
        INCLUDE "p16f628.inc"         (Nada que decir, la incluye el compilador)
        INCLUDE <BUS_I2C_.INC>        (librería de manejo del bus I2C)
        INCLUDE <24LC256.INC>         (librería de manejo de la EEPROM)

           END    (FIN ☺ )
```

LAS LIBRERIAS

Como pueden ver éste código solo utiliza tres librerías que son:

- p16f628.inc ; La incluye el compilador no hay nada especial.
- BUS_I2C_.INC, es la librería que permite el manejo del bus I2C de la EEPROM
- 24LC256.INC, es la librería que maneja la EEPROM.

En realidad el bucle principal de programa no hace uso directo de la librería del bus I2C, es la librería de manejo de la EEPROM la que llamas a las funciones adecuadas Para leer cada bit/byte según el estándar del I2C.
En Internet hay muchas librerías para manejar bus I2C y EEPROM de muchos tipos y modelos, el usar librerías me permite con pocas modificaciones en el bucle principal usar distintos modelos.

¿AUDIO COMPRIMIDO?

Este PIC no tiene suficiente potencia para manejar audio comprimido, como pueden ver está usando toda la velocidad posible del bus I2C hay muy pocas líneas de código leemos un byte, lo mandamos al puerto y volvemos a leer otro byte, aún así el audio tiene poco ancho de banda, los recursos son muy limitados, no es posible sacar más rendimiento a un hardware tan sencillo.
No hay modulación delta ni compresión de ningún tipo, es todo muy sencillo e intuitivo.

ESQUEMA ELECTRÓNICO DEL PROYECTO

